UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/608,985	06/26/2003	Amitabh Srivastava	3382-64710	6401

26119 7590 03/26/2008
KLARQUIST SPARKMAN LLP
121 S.W. SALMON STREET
SUITE 1600
PORTLAND, OR 97204

EXAMINER	
VU, TUAN A	
ART UNIT	PAPER NUMBER
2193	
MAIL DATE	DELIVERY MODE
03/26/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 10/608,985	**Applicant(s)** SRIVASTAVA ET AL.	
	Examiner Tuan A. Vu	**Art Unit** 2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04 January 2008*.

2a)☒ This action is **FINAL**. 2b)□ This action is non-final.

3)□ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-36* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)□ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-36* is/are rejected.

7)□ Claim(s) _____ is/are objected to.

8)□ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)□ The specification is objected to by the Examiner.

10)□ The drawing(s) filed on _____ is/are: a)□ accepted or b)□ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)□ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)□ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)□ All b)□ Some * c)□ None of:

1.□ Certified copies of the priority documents have been received.

2.□ Certified copies of the priority documents have been received in Application No. _____.

3.□ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) □ Notice of References Cited (PTO-892)

2) □ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) □ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____.

4) □ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .

5) □ Notice of Informal Patent Application

6) □ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 1/04/2008.

As indicated in Applicant's response, claims 1, 15, 18, 21, 25, 27, 29, 32 have been amended. Claims 1-36 are pending in the office action.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 21, 25 are provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 2, 14 of copending Application No. 11,330,053 (hereinafter '053), further in view of Srivastava et al., "Effectively Prioritizing Tests in Development Environment", February 2002, MSR-TR-2002-15, Publisher: *Association for Computing Machinery, Inc* (hereinafter Srivastava). Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following conflicts.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

As per instant claim 21, '053 claim 2 also recites prioritizing test (which include performing tests) based on indicators indicating which portions are impacted by tests, portions including plurality of blocks of binary code each *having entry and exit points*; wherein indication (i) indicates ones of the plurality of blocks are modified, hence this reads on marking changed logical abstractions; but '503 does not recite:

determining dependency information about binary files, propagating such information to determine subsystem and system dependency, marking changed and unchanged logical abstraction to prioritize tests.

Srivastava in a environment for obtaining list prioritized tests (see pg. 5-7) to perform tests, similar to '053; and not only discloses using previous test results or data being marked to feed into new prioritized test, but also discloses binary files analysis to block system and subsystem, recording changes thereto in a propagation analysis hashing (L column, 4th para, pg. 3), and further obtaining dependency information (e.g. *along with ... coverage information* – L column 1st para, pg. 2; *changes ... propagated* – R column, 5th para, pg. 2; *list of modified blocks* – top para, R column. pg. 3; *marked as old block, impacted blocks, matching blocks* -- pg. 3, R column) so as to propagate it into graph elements in order to mark intra-procedural or interprocedural changes so to determine what are new block or unchanged blocks or potential impacted new block in view of a subsequent test coverage (see Fig. 1, Fig. 2, pg. 3-4); and producing prioritized lists (see top pg. 3).

It would have been obvious for one skill in the art to implement the prioritized tests by instant claims 21, 25, and 29 so that ‘053 steps of creating list of prioritized tests recording modified or new blocks with respect to a given test version, which is analogous to Srivastava from above, be implemented to include using test results into a test optimization instance and effecting for which a dependency information determination and propagation via graph and hashing algorithm (see Srivastava) to enable marking of system and subsystem of abstractions in binary files as taught by Srivastava, because this enable appropriate marking as suggested in (i) leading to optimization on how to reduce paths in recurring tests for a given test version based on dependency information propagated in target binary files as Srivastava discloses (see Fig. 2).

As per instant claim 25, ’053 claim 14 also recites prioritizing test (which include performing tests) of a target software having plurality of blocks being executed, based on indicators indicating which portions are impacted by tests. But '053 does not explicitly recite block having entry and exit point; nor does ‘053 recite storing dependencies and propagated dependencies; marking changes and propagating marked changes for prioritizing test based test coverage of such propagated marked changes. ’053 suggests changes being marked via using of indicators – as in (i) -- within the target code to collect indication as to determine which blocks have been impacted by the instances of software binaries tested. Based on the tool by Srivastava, where in block of code are identified as tree node with entry and exit points for enabling propagation affected one block relative to another (see inter-procedural, intra-procedural from above), the marked changes being propagated as suggested above and Srivastava's tree propagation of impacted blocks, the rationale as to render obvious ‘propagating marked changes’ for prioritizing test coverage, as set forth above, will be incorporated herein.

Claim Objections

4. Claim 1 is objected to because of the following informalities: the ‘responsive to the request sent via' appears to be improper usage of what the concept behind 'responsive to'. In commonly accepted meaning as for 'response to’ reception of a request, what is required would be an associated responsive action (yielding a delivered output) performed by the receiving end opposite the originator of the request. As claimed, ‘receiving … a dependency collection' cannot be construed as responsive action done by the service that receives the original request, but rather by the requesting end, which renders usage of the phrase “receiving ,responsive to ..." improper. Thus, 'responsive to' would be understood more like as a consequence of. Based on the Specifications (see pg. 18, li. 16-25), the receiving is associated with a human being sending a request via a tool API. The final ‘receiving’ step for lack of specificity and clarity from the Disclosure, would be interpreted as receiving within the framework by the above tool, such that 'receiving', 'requesting' then 'receiving' as sequenced in the claim amount to actions taken by the tool itself. That is, in support for clarifying on ‘responsive to’, no real and concrete action is depicted (by the claim as a whole) as to whether any action is being executed responsive to (emphasis added) the above *requests*, e.g. no action actually taken to put together a collection of dependency data.

5. Further, the phrase ‘indicating a chosen’ as recited in ‘…abstraction indicating a chosen structural level of complexity…’ (claim 1, line 7) is objected to because it implicates a human action. According to the Disclosure, this choice is depicted as a *request* including developer input or a manager's intervening instruction such as *indicating*; and this cannot establish that the inventor possesses a tool/process that performs both ‘requesting' and ‘receiving’. In regard to

request and a *choice* on ‘level of complexity of abstraction’, the Specifications (e.g. Example 4: pg. 13, li. 16-24; Example 9: pg. 18, li. 21-25) exposes the ‘receiving’ step (in conjunction with *request*) in a way that is not part of the invention but rather pertains to a user action or initiative. The method has to be recited as to reasonably convey a mechanism performed on its own capacity as opposed to relying on user's initiative; and if sustained, the impropriety can and will lead to a statutory USC 101 type of deficiency (emphasis added). Appropriate correction is strongly recommended.

6. Claim 27 is also objected to for reciting ‘receiving’, ‘requesting’, ‘indicating a *chosen*', and ‘receiving, *responsive to the request*’ a dependency collection as in claim 1; i.e. because of the improper use of a terms that renders the context of the claim non-commensurate with what the invention is all about.

Claim Rejections - 35 USC § 112

7. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

8. Claims 1-20, 27-28 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. That is, claims 1, 27 are rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential steps, such omission amounting to a gap between the steps. See MPEP § 2172.01. The omitted steps are: there is no step being performed when a request comprising *indicating a chosen* level of complexity is sent via an API for a *dependency collection* to be retrieved from the method. As described above in the Claims Objections in regard to user '*indicating* a chosen structural', the request in light of the

Specifications enables a *manager* or the *user* to input (i.e. *indicating*) a certain block of code pertinent to some selected level of the binary hierarchy (Example 4: pg. 13, li. 16-24; Example 9: pg. 18, li. 21-25); hence the ‘requesting’ , ‘indicating’ then receiving, 'responsive to the request' cannot be construed all together, as steps performed by the method, but rather by the very user of the tool and its API; further, the claim lacks actual step taken by the method to generate *data dependency* needed for it to be sent back to the requesting user or entity. In whole, as construed in the analysis made above (see Claims Objections), the claim amounts to human interface whereby request is sent from a user input and initiative, via a API; notwithstanding the fact that *data dependency* amounts to mere ‘descriptive material’ which is received back in response to said user (Note: descriptive material cannot be equated to actual action performed by the method claim). Method claims should depict process steps by which the invention performs on data to yield data, and in light of the Specifications and the language of the claim, there is no step of performing any creating of data leading to a result received by the 'receiving step'. Scanning the disclosure for the term 'request', one cannot see any action taken by the tool (e.g. receiving, responsive to) like ‘requesting' or ‘indicating’ (in conjunction with the user’s choice) thereby generate data (e.g. based on user selection of block as input into this request). It is deemed based on the Specifications, that the claim is insufficient in terms of action details or support that would enable one of ordinary skill in the art to see what (metes and bounds) the invention actually does or is capable of transforming or creating (e.g. for 'data dependency' recited as mere descriptive material, to be outputted by the claimed method). Some actions taken and performed by the ‘method claim' is clearly lacking or omitted based on what appears to be mere user’s sequence of steps. An invention has to perform actions with cohesive structural relationship to yield a tangible useful result; here, the claim describes user interaction based on a choice specificity from the user, no data actually transformed, no process depicted to effectuate such transformation. The phrase 'indicating a chosen level' will be treated accordingly. The impropriety if not corrected would lead to a form of non-statutory subject matter (emphasis added).

Claim 27 is also rejected for reciting 'receiving', 'requesting', 'indicating a *chosen*', and 'receiving, *responsive to the request*' a dependency collection as in claim 1; hence cannot enable one of ordinary skill to reasonably construe (in light of the Specifications) that the invention is capable of performing the *requesting*, *indicating* and *receiving* , as well as responding to all of which with taken actions like data transformation on its own in a <u>concrete</u> manner independent from on any user.

Claims 2-20, 28 are rejected for not curing to the above lack of definite support for the recited steps.

Claim Rejections - 35 USC § 102

9. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

10. Claims 21-26, 29-31 are rejected under 35 U.S.C. 102(b) as being anticipated by Srivastava et al., "Effectively Prioritizing Tests in Development Environment", February 2002, MSR-TR-2002-15, Publisher: *Association for Computing Machinery, Inc.*, pp. 1-10 (hereinafter Srivastava_2).

As per claim 21, Srivastava_2 discloses method comprising:

determining dependency information about binary files (e.g. *along with ... coverage information* – L column 1st para, pg. 2; *changes ... propagated* – R column, 5th para, pg. 2; *list of modified blocks* – top para, R column. pg. 3; *marked as old block, impacted blocks, matching blocks* -- pg. 3, R column - Note: using previous results or changes in files, and determining blocks for marking and marking insides binary files to find out about subsequent path on how the block test coverage is to be done reads on determining dependency information based on which to apply optimization algorithm upon runtime resources), dependency information comprising entry and exit points for binary files (see Old binary New Binary – Fig. 1, pg. 3; Version 1, Version 2 - Table 1, pg. 4 -Note: any binary file version for execution purpose inherently includes entry point at which code starts and point at which the file terminates);

propagating dependency information about binary files to determine subsystem dependency information comprising entry and exit points for a subsystem of the system (Note: information yielded from previous tests – see Table 1, Table 2, Table 3 - regarding how to cover new tests – see Fig. 2 -- based on impacted blocks reads on information including tree-like blocks of procedure or basic blocks with entry and exit points – *heuristic ...successor predecessor blocks* – pg. 3, R column ; *interprocedural graph ... immediate predecessors* – pg. 3, bottom half R col; *likely to be taken* – R column, bottom pg. 3);

propagating the subsystem dependency information to determine system dependency information (e.g. *changes ... propagated* – R column, 5th para, pg. 2 – Note: marking from using information changes to affected parts of a source binary as this is scanned under analysis reads on propagating this information determination or change information and use it for marking of blocks for a given file version, i.e. subsystem of system or dependency of system – see Table 1, Table 2, pg. 4 ; *interprocedural graph … immediate predecessors* – pg. 3, bottom half R col.) comprising entry and exit points (see Old binary New Binary – Fig. 1, pg. 3; Version 1, Version 2 - Table 1, pg. 4) for the system;

marking changed logical abstractions; marking unchanged logical abstractions dependent on marked changed logical abstractions in other subsystems (*marked as old block, impacted blocks, matching blocks* -- pg. 3 R column);

comparing test coverage to marked changed logical abstractions and to marked unchanged logical abstractions (e.g. *heuristic …successor predecessor blocks* – pg. 3, R column ; *likely to be taken* – R column, bottom pg. 3; Fig. 2 – Note: based on marking information between old and new block, and applying heuristic thereto in order to predict how to skip path reads on comparing based on marked of changed and unchanged); and

prioritizing tests based on maximum test coverage of marked changed logical abstractions and marked unchanged logical abstractions (e.g. Fig. 2);

and performing the prioritized tests according to test priorities to produce test results (top para, L col., pg. 2; Figs. 8, 10, pg. 6-7).

As per claims 22-23, Srivastava_2 discloses coverage comprises tests for one subsystem (e.g. sequence Set – Fig 2 – Note: block set being tested based on weight of impacted blocks reads on subsystem of binary blocks), a subsystem among a plural subsystems.

As per claim 24, Srivastava_2 discloses the test coverage comprises tests for plural subsystems and maximum test coverage is considered for marked changed logical abstractions and marked unchanged logical abstractions (pg. 4, L column; Fig. 2 – Note: impacted blocks based on marking of old and new blocks reads on test coverage for marked changed and unchanged – see R col., pg. 3) for said plural subsystems.

As per claim 25, Srivastava_2 discloses a computer-based service comprising **means for** performing the same steps as recited in claim 21, namely:

determining binary dependencies for a defined system (e.g. *along with ... coverage information* – L column 1st para, pg. 2; *changes ... propagated* – R column, 5th para, pg. 2; *marked as old block, impacted blocks, matching blocks* -- pg. 3 R column) comprising entry and exits points for a defined system (e.g. Old binary New Binary – Fig. 1, pg. 3; Version 1, Version 2 - Table 1, pg. 4 -Note: any binary file version for execution purpose inherently includes entry point at which code starts and point at which the file terminates);

propagating binary dependencies to identify binaries dependent on binaries in other subsystems and storing determined and propagated dependencies (*changes ... propagated* – R column, 5th para, pg. 2 - Note: marking from using information changes to affected parts of a source binary as this is scanned under analysis reads on propagating this information determination or change information and use it for marking);

marking changes and propagating marked changes using the determined and propagated dependencies; and prioritizing tests based on test coverage of marked changes and propagated marked changes;

performing prioritized tests according to the prioritizing;

all of which steps having been addressed in claim 21.

As per claim 26, Srivastava_2 discloses marking proposed changes (e.g. *proposed* – top para, R column, pg. 2; *new version ... likely to be covered by a existing test* – 3rd para, R column, pg. 3 – Note: applying a test to a new code reads on proposed changes to an current test scenario).

As per claim 29, Srivastava_2 discloses computer system comprising:

a processor coupled to volatile and nonvolatile memory; binary files stored in memory (see Binary - Fig. 1); software stored in memory comprising computer executable instructions for:

determining dependency information for binary files comprising entry and exit points for the binary files (re claim 21),

propagating dependency information to determine subsystem dependency information comprising entry and exit points for a subsystem (refer to claim 21) of a system, and

propagating subsystem dependency information to determine system dependency information comprising entry and exit points (see claim 21) for the system;

marking logical abstractions changed from a previous version (e.g. *Old Binary, New Binary* - Fig .1, R column pg. 3);

propagating marked changes according to the dependency information comprising marking unchanged logical abstractions dependent on marked changes in other subsystems (refer to claim 21);

comparing test coverage to marked changed logical abstractions and to marked unchanged logical abstractions; and prioritizing tests based on maximum test coverage of marked changed logical abstractions and marked unchanged logical abstractions (refer to corresponding rejection in claim 21).

As per claims 30-31, Srivastava_2 discloses maximum test coverage is based on the total number of marked changed and marked unchanged logical abstractions touched (e.g. Fig. 2; L column, pg. 4) by a test system wide (Note: covering of subset or test sequence per iteration reads on system wide); wherein maximum test coverage is based on the sum of the total number of marked changed logical abstractions in a first subsystem touched by a test, and marked unchanged logical abstractions touched by the test in the first subsystem, wherein the marked unchanged logical abstractions depend on marked changed logical abstractions in other subsystems (e.g. pg. 3, R column 2nd para, bottom para, pg. 3 to L column, pg. 4; ch. 4-5, pg. 4-5).

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. Claims 1-20, 27-28, 32-36 are rejected under 35 U.S.C. 103(a) as being unpatentable under Srivastava et al, “Vulcan: Binary transformation in a distributed environment”, April 2001, Technical Report MSR-TR-2001-50, pp. 1-12 (hereinafter Srivastava), in view of Srivastava et al., “Effectively Prioritizing Tests in Development Environment”, February 2002, MSR-TR-2002-15, Publisher: *Association for Computing Machinery, Inc.*, pp. 1-10 (hereinafter Srivastava_2).

As per claim 1, Srivastava discloses a method comprising:

receiving a system definition (e.g. *PDB file, meta-information* – sec 3.1,pg. 6, R col) and a request for dependency information (Fig. 2 – Note: submission of input into a API reads on request using Vulcan), the system definition comprising metadata describing a system at a level of abstraction (e.g. *PDB file, meta-information* – sec 3.1,pg. 6, R col to pg. 7, top L col);

requesting, via an application programming interface, a dependency collection, by sending to the application programming interface a system dependency creation request (e.g. Fig. 2) comprising the received system definition and a dependency request (sec 2.1 pg. 5 – Note: input form of abstraction submitted in a API to yield output level via externalizing – see Fig. 2 -- the received abstractions; *Vulcan API* - Appendix, pg. 11 discloses *via a application programming interface*) comprising a target logical abstraction;

receiving, responsive to the requests sent via the_application programming interface, a dependency collection (Fig. 3; *Vulcan API, chaining* – Appendix A, pg. 11) for the target logical abstraction comprising logical abstractions in one or more dependency chains with the target logical abstraction (e.g. *six basic abstractions, list of Basic Block, CFG, DFG* – sec. 2.2 pg. 5, R col; Control flow based Basic block – Fig. 3, pg. 7),

wherein the dependency collection comprises logical abstractions outside of a subsystem for the target logical abstraction (see *abstraction ...different component ...heterogeneous* – pg. 4, R col; Fig. 1 – Note: binary file version as a system of procedure subblocks reads on file version external to another system and also external to respective sub-blocks or procedures of another file version or system -- see Table 3, pg. 14); wherein the dependencies chains represent chained dependencies between subsystems (e.g. *code blocks, procedures* – Table 3, pg. 14; sec 2.2, pg. 5 ; Control flow based Basic block – Fig. 3, pg. 7); and displaying a representation of the collection (e.g. *externalized ...for... transformation, testing debugging* – pg. 4, R col. bottom; *abstract representation .. queried analysis modified for instrumentation* – sec 2.1, pg. 5 L col – Note: Win32, Microsoft system and IA 64 architecture based tool with externalization of abstraction for debugging, modification and optimizing reads on displaying).

Srivastava does not explicitly disclose dependency request comprising target logical abstraction indicating a chosen structural level of complexity of system described by the system definition. In a framework analogous to Srivastava (see Srivastava: sec 5 pg 9) for analyzing possible changes to binary files across systems including, instrumentation, recompiling to detect risks from replaying previous version thereof and identifying stale profile, Srivastava_2 discloses using Vulcan/Echelon in conjunction with BMAT (as Srivastava) and execution for detecting impacted of block of codes; wherein code representation as information collected from the GUI tool in support for impact testing includes blocks (see Fig .1; sec 3.1 pg. 3); wherein Srivastava_2 teaches a level of granularity as to how much coverage can test sequences be predetermined (see Fig 3-4; pg. 5 R column) and a predetermined set t to cover a given block deemed to be of some impact value (see *ImpactedBlkSet, Coverage (t)* – Fig. 2, pg. 4) the suggestion that data dependency based on a certain level of blocks to cover is based on developer's directives. It would have been obvious for one skill in the art at the time the invention was made to implement the ordering of test as by Srivastava so that dependency information obtained from Echelon representation of binary versions to test (see sec 3.1) is derived from the user's specification of a number of granularity as suggested above, based on impacted set identified via repetitive stages of benchmark; because this narrowing of the level of blocks to investigate via Echelon, as purported by Srivastava_2 or Srivastava's BMAT (Table 4, 5) would alleviate repeating of test to those blocks of further (e.g. of no impact) interest, which is consistent with the desired results shown in Srivastava_2 learning of impacted blocks (see Figure 9-11, pg. 7)

As per claims 2-3, Srivastava discloses wherein the system definition is received as a file identification (e.g. *meta-information ... PDB file* – sec 3.1, pg. 6, R col); wherein the system definition is received via a graphical user interface (refer to Vulcan debug tool with display as per claim 1)

As per claims 4 and 7, Srivastava discloses wherein the target logical abstraction is an unchanged logical abstraction (e.g. *Basic Block, CFG, DFG* – sec 2.2. pg. 5 R col.), wherein the target logical abstraction is a changed logical abstraction (e.g. *instrumentation interface ...inserted ... Basic Block* – pg. 6, L top col.).

As per claims 5-6, Srivastava discloses wherein the dependency collection comprises logical abstractions dependent on (Fig. 2; sec 2.1, pg. 5) the target logical abstraction; wherein the dependency collection comprises logical abstractions (e.g. Fig. 2; Intel x86, Risc, Compaq Alpha – pg. 5, R col.) on which the target logical abstraction depends.

As per claims 8-9, Srivastava discloses wherein the target logical abstraction comprises a basic block, a procedure, or a binary file (e.g. Procedure, Basic block, sec 2.2, pg. 5); wherein the dependency collection comprises a basic block logical abstraction (e.g. pg 5, R col).

As per claim 10, refer to claim 8, wherein the dependency collection comprises at least one of a procedure logical abstraction or a binary file logical abstraction.

As per claim 11, Srivastava discloses wherein the dependency collection comprises a named object logical abstraction (e.g. *Component* – pg. 5, top R col) or a node logical abstraction (e.g. CFG, DFG - pg. 5, R col.).

As per claim 12, Srivastava discloses wherein the representation of the collection comprises a number of affected logical abstractions (e.g. instrumentation …inserts probes … reorders basic blocks – Fig. 6, pg. 7 L col. top).

As per claim 13-15, Srivastava discloses wherein the representation of the collection comprises a graphical presentation of a dependency chain (graphical display: refer to claim 1; *Basic Block, CFG, DFG* – sec 2.2. pg. 5 R col) wherein the representation of the collection comprises a list of logical abstractions (e.g. *list of Basic Blocks* -pg. 5, R col); and further comprising displaying a representation of the collection comprises a graph of logical abstractions (re claim 1).

As per claim 16, Srivastava discloses wherein the dependency collection further comprises logical abstractions (e.g. *System*, *collection of Programs, list of components, Component consisting ... symbol, data blocks* – pg. 5, top R col) inside the logical abstraction's subsystem.

As per claims 17-18, Srivastava does not explicitly disclose wherein the logical abstraction comprise a proposed change, and displaying a metric for said proposed change risk, wherein said risk comprise a number of abstractions affected by the proposed change in relation to the number of abstractions. In a framework analogous to Srivastava (see Srivastava: sec 5 pg 9) for analyzing possible changes to binary files across systems including, instrumentation, recompiling to detect risks from replaying previous version thereof and identifying stale profile,

Srivastava_2 discloses using Vulcan in conjunction with BMAT (as Srivastava) and further teaches code execution for detecting impacted of block of codes in view of the number of blocks being observed (e.g. Fig. 4-5, pg. 5 – Note: percentage of impacted blocks over total sequences reads on display of proposed risk changes in view of proposed changes - *proposed* – top para, R column, pg. 2); i.e. percentage reads on displaying a change risk metric. Hence, in view of the same environment using instrumentation/profiling for binary changes analysis applicable to multi-system application version improvement, it would have been obvious for one skill in the art at the time the invention was made to enable the abstraction representation in Srivastava's debugging and dynamic instrumentation tool such that it include the analysis of binary code as in Srivastava_2 so that a impact percentage among the level of abstraction being analyzed by the tool be marked as a displayed metric so to support whether the whole code incur sufficient risk so that the software development team can decide to rebuild the target program binary code based on such percentage metric as endeavored by Srivastava_2's large scale prioritization of applications and (see Srivastava_2: Abstract, Introduction, pg. 1;), which is commonly contemplated by Srivastava for scalability within larger commercial application including optimizing via determination on how much to recompile (see Introduction, pg. 3; Partial compilation -sec 6.1, pg. 9).

As per claim 19, Srivastava_2 discloses wherein the relation is further adapted with a logarithmic function (see Fig. 5, pg. 5); hence the rationale so to display a metric within such logarithmic function of binary code behavior would have been obvious in view of the same tool and endeavor by Srivastava and Srivastava_2.

As per claim 20, refer to claim for computer-readable medium having executable instructions for performing the method of claim 1.

As per claim 27, Srivastava discloses a computer-readable medium (see claim 20) having executable instructions performing a method comprising:

creating a system definition comprising metadata describing a system at a level of abstraction_in response to receiving graphical user interface input;

receiving a dependency information request via graphical user interface input;

requesting via an application programming interlace exposed by a dependency framework, a dependency collection, by sending to the application programming interface a system dependency creation request comprising the system definition, and a target logical abstraction identifiable from the dependency information request (refer to claim 1); and

receiving, responsive to the requests sent via the_application programming interface, a dependency collection for the target logical abstraction comprising logical abstractions in one or more dependency chains representing chained dependencies between subsystems (*code blocks, procedures* – Table 3, pg. 14; sec 2.2, pg. 5; graphical display: refer to claim 1; *Basic Block, CFG, DFG* – sec 2.2. pg. 5 R col) with the target logical abstraction;

wherein the dependency collection comprises logical abstractions outside of a subsystem for the logical abstraction; all of which limitations having been addressed in claim 1.

Srivastava does not explicitly disclose *indicating a chosen structural level of complexity of the system*; but this limitation has been addressed in claim 1.

As per claim 28, refer to claim 16.

As per claim 32, Srivastava discloses a user interface service comprising:

means for accepting a system definition comprising metadata (*PDB file, meta-information* – sec 3.1,pg. 6, R col) describing a system at a level of abstraction (sec 2.2 pg. 5) and indicating_binary files (e.g.*from the application binaries* - sec 2.1 pg. 5) in plural subsystems;

means for accepting an indication (sec 2.2, pg. 5; Fig. 2 – Note: input into API for requesting abstraction representation by Vulcan reads on indication) of a target logical abstraction; and

means for displaying dependency relationships (e.g. *externalized ...for... transformation, testing debugging* – pg. 4, R col. bottom; *abstract representation .. queried analysis modified for instrumentation* – sec 2.1, pg. 5 L col – Note: Win32, Microsoft system and IA 64 architecture based tool with externalization of abstraction for debugging, modification and optimizing reads on displaying) between the target logical abstraction and a set of logical abstractions in binary files between two or more of the plural subsystems (*code blocks, procedures* – Table 3, pg. 14; sec 2.2, pg. 5).

Srivastava does not explicitly disclose *indicating a chosen structural level of complexity of the system*; but this limitation has been addressed in claim 1.

As per claims 33-34, Srivastava's tool display is not disclosed as comprising means for displaying a proposed change risk and means for displaying a change risk metric. But these limitations have been addressed in the rationale set forth for claims 17-18 respectively.

As per claim 35, Srivastava's tool and user interface is not disclosed as comprising means for displaying a graph of relative risk for plural subsystems. But the graph with metrics indicating a percentage of impact leading to evaluation of risk and subsequent determination for rebuilding binary has been addressed in claims 17-18, as obvious in view of the similar endeavor and purports by both Srivastava and Srivastava_2.

As per claim 36, Srivastava discloses means for displaying test coverage evaluation (pg. results (e.g. *code coverage ... test* – pg. 9, R col. top; Fig. 7-10).

Response to Arguments

13. Applicant's arguments filed 1/04/2008 have been fully considered but they are either moot or not persuasive. Following are the Examiner's observation in regard thereto.

35 USC § 102 under Srivastava:

(A) Applicants have submitted that as amended claims 1, 27 and 32 recite 'target logical abstraction indicating a chosen structural level' and 'dependent chains represent chained dependencies between subsystems' and assertions to the effect that Srivastava does not '… indicates a chosen structural level of complexity…' (Appl. Rmrks pg. 11-12). The grounds of rejection now address particularly the scope of the claims as well as the specifics of the amended claims according to their derived merits. The arguments would be moot because they are no longer commensurate with the previous grounds of rejection effective prior to these recent amendments.

(B) Applicants have submitted that Srivastava does not teach 'abstract representation' of a program, and that Vulcan's program representations are not showing 'dependencies or chained dependencies' as recited (Appl. Rmrks pg. 13, top half). Control flow between block or basic blocks appear to be what one of ordinary skill in the art construes as blocks representative of source code being chained in a dependency type of graphical tree, where control flow can be illustrated between upper and lower block being linked by such graph. Srivastava has been recited with proper sections where basic blocks are chained to depict abstracted representation of code in a manner that such dependency between blocks are displayed (refer to rejection). The argument is not persuasive.

(C) Applicants have submitted that peephole optimization with branch chaining cannot teach or suggest 'chained dependencies' of claim 1 (Appl. Rmrks bottom pg 13). This has to be referred to section B above. In regard to claims 27 and 32, new grounds of rejection has render Applicants remarks (Appl. Rmrks pg. 14, top) moot.

35 USC § 102 under Srivastava_2:

(D) Applicants have submitted that under Srivastava_2, based on the added changes to the claims 21, 29, Srivastava_2 does not teach or suggest that 'propagate dependency information' in Srivastava_2 include "... entry and exit points ...' as recited (Appl. Rmrks pg. 15, bottom). The rejection has addressed 'entry and exit points' with proper citations according to this new grounds of rejection which are deemed necessitated by the addded limitations. The language recited as 'propagate dependencies' is absolutely silent in terms of what exactly 'dependencies' constitutes of, if not for the above entry and exit points; nor does the claim describe how this propagating has been accomplished in specific detail including what precise piece of information has been transported from one subsystem block to the next. The instant Specifications mentions about changes affecting previously executed blocks, wherein marking help identifying such changes within examining of a control flow, and such teaching fall under the endeavor of BMAT based on which Srivastava_2 analyzes binaries version differentials using CFG. Based on the above, it is not clear how the claim by just reciting 'propagating' --as in a vacuum-- can suddenly be such that it clearly distinguishes (as alleged by Applicants) from Srivastava's identifying of affected code blocks via re-analyzing of impacted blocks from re-executed version of binaries using a CFG and a BMAT, when ‘dependencies' as claimed amounts to information including entry and exit points. Applicants emphasizes on the ‘entry and exit points’ being key in giving weight to this ‘propagating dependencies’ or propagating information. As such, *propagating* (dependencies) could be merely interpreted as joining blocks of code representing a control flow of a source program such that upper blocks are joined with lower blocks, using exit point from one node and entry point into another; and this is not novel with respect to Srivastava_2’s CFG being generated. Further, one of ordinary skill in the art would find hard how said ‘propagating’ limitation can be any more than just including information having ‘entry and exit points' for code abstracted blocks. Besides, *propagating* entails information being conveyed or transported from one point to another, and the claim does not remotely structure its language to reasonably depict the minimal underlying mechanism for the concept of ‘propagating’. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference.

(E) Applicants have submitted that Srivastava_2 does not teach ‘marking unchanged logical abastraction dependent on marked changed … in other subsystems’ because there is no indication that Srivastava_2 use ‘dependencies’ (Appl. Rmrks pg 16, top half). The marking of affected blocks or unaffected blocks has been depicted and cited in the Office Action. The ‘dependencies’ term is treated as block lay out or procedural graph, and Srivastava_2 has been deemed fulfilling the marking of blocks when identifying which has not be impacted by a re-run of a modified version using Echelon and interprocedural analysis (refer to: R column top, p 3).

Changed and unchanged dependencies amount to identification of marked changed blocks versus marked unchanged blocks in use of Echelon. The language of the claim (e.g. claim 21, 25) does not tie 'propagating dependencies' with 'marking changed ... abstractions'; nor does the claim convey how 'changed abstractions' get implicated in relation to (emphasis added) the propagating; nor is there any remote structural relationship between 'system', 'subsystems', 'binary files' and 'changed logical abstractions' in light of the propagating and 'test coverage' to marked changed abstractions (e.g. how can *coverage* be related to abstractions, not knowing if abstractions represent anything related to binary files? How can changes marked when only 'dependency information' has been the only step achieved, not knowing if the changes correlates any abstraction with any of the above dependencies by any stretch of imagination; are the marked -changed or unchanged- abstractions any distinct from the counterparts within binary system and subsystems implicated with the dependency information? Are changes being changes to the code within the binary files or to the abstraction? If so, how can an abstraction collect/integrate such binary changes or markings, i.e. are the markings (to the abstractions) already incorporated by the time these abstractions are collected or introduced? Whose abstractions are these all about? How does a test cover a plurality of abstractions when there is no mention of **test being executed** from a point where only information is gathered). The Specifications cannot be read into the claim, and the claim is clearly lacking cohesive or at least a meaningful relationship or constructive rationale to convey some cooperation among all elements recited. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference.

Using best effort to interpret the claim language, the Office action has derived some interpretation and based on which has deemed that the marking by Srivastava_2 using of Echelon to lay out code blocks from comparing executed binary version has fulfilled the above language. The argument is considered not sufficient to overcome the rejection.

(F) Applicants have submitted that claims 25, 29, 17-19, 33-35 are also patentable in view of Srivastava_2 not able to disclose 'dependency information' in conjunction with ‘test coverage’ of marked changed/unchanged abstractions (Appl. Rmrks pg. 17-18). Refer to section E.

Double Patenting Rejection:

(G) Applicants have submitted (Appl. Rmrks pg. 19) that '053 cannot render obvious the “determine dependency information ... " limitation in view of Srivastava’s deficiency. The argument is not persuasive to the extent that Srivastava will remain outstanding and effective with respect the above arguments.

In all, the claims stand rejected as set forth in the Office Action.

Conclusion

14. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

March 22, 2008